

Relazione Progetto Basi di dati - Andrea Serrano 0000988271 - AnswerHUB

1) Raccolta/Analisi dei requisiti

1.1) Testo completo delle specifiche sui dati

Si vuole realizzare la piattaforma EFORM per la gestione di sondaggi online (liberamente ispirata al servizio di Google Form). La piattaforma consente di popolare depositi di domande a risposta aperta/chiusa e, a partire da questi, di generare sondaggi/quiz da sottoporre ad utenti registrati della piattaforma, in forma anonima o meno. La piattaforma supporta inoltre la registrazione di nuovi utenti/aziende e la gestione delle risposte dei sondaggi.

SPECIFICA DELLA PIATTAFORMA. La piattaforma EFORM consente la creazione/gestione di sondaggi online. Ogni sondaggio dispone di un codice (univoco), un titolo, una data di creazione, una data di chiusura, uno stato, un numero massimo di utenti partecipanti (maxutenti). Lo stato è un campo ENUM con valori: APERTO o CHIUSO. Ogni sondaggio è composto da una o più domande: ogni domanda dispone di un id (intero auto-incrementale), un testo, un'eventuale foto, un eventuale punteggio. Le domande appartengono a due tipologie (e solo ad esse): domande aperte o chiuse. Le domande aperte dispongono anche di un numero massimo di caratteri (per la risposta). Le domande chiuse dispongono di un certo numero (>1) di opzioni. Ogni opzione dispone di un numero progressivo (univoco, ma solo per quella domanda) e di un testo. La stessa domanda può apparire in più sondaggi. Ogni sondaggio dispone di un dominio che identifica il macro-argomento del sondaggio stesso. Ogni dominio consiste in una singola parola chiave -univoca- (es. "cinema") e di una descrizione (es. "sondaggi sul gradimento di film, attori e registi"). Si vogliono gestire le informazioni degli utenti registrati nella piattaforma: ogni utente dispone di email, nome, cognome, anno e luogo di nascita e di un campo totalebonus (maggiori dettagli a seguire). Ogni utente può manifestare il proprio interesse verso zero, uno o più domini tra quelli presenti sulla piattaforma. Alcuni utenti (ma non tutti) sono utenti premium o utenti amministratori. Gli utenti premium dispongono anche di un campo data inizio abbonamento, data fine abbonamento e costo, e di un campo #numsondaggi (ridondanza concettuale). In aggiunta, sulla piattaforma possono registrarsi aziende: ogni azienda dispone di un codice fiscale, un nome, una sede, un indirizzo email. Solo utenti premium o aziende possono inserire domande e creare sondaggi: un sondaggio è associato ad un solo creatore (utente premium o azienda). Ad ogni sondaggio è associata una lista di utenti (di qualsiasi categoria) della piattaforma che vi partecipano, a valle dell'accettazione di un invito. In particolare,

occorre tenere traccia degli inviti: ogni invito è relativo ad un solo sondaggio, è inviato ad un utente registrato della piattaforma, e dispone di un codice univoco e di un esito (accettato o rifiutato). La lista degli utenti da invitare deve essere gestita in maniera differente, a seconda che il sondaggio sia creato da un utente premium o da un'azienda. Nel caso dell'utente premium, quest'ultimo seleziona gli utenti da invitare mediante l'interfaccia della piattaforma. Nel caso del sondaggio dell'azienda, la lista viene generata in maniera automatica, scegliendo CASUALMENTE gli utenti da invitare tra quelli i cui interessi includono il dominio del sondaggio stesso. Inoltre, i sondaggi creati dall'azienda devono essere gestiti in forma anonima, quindi i dati dei partecipanti (es. email, nome, cognome) NON devono essere visualizzati sull'interfaccia della piattaforma. Tuttavia, in entrambi i casi (sondaggi creati da utenti premium o da azienda), la lista degli inviti deve essere salvata su database. Ogni qualvolta un utente accetta un invito ad un sondaggio, il suo campo totalebonus deve essere incrementato di un valore pari a 0.5. Ogni utente che accetta un invito può inserire le risposte per ciascuna domanda del sondaggio. Si vuole tenere traccia delle risposte, sia di quelle a domanda aperta (un campo testo) sia di quelle alle risposte chiuse (la lista delle opzioni selezionate tra quelle disponibili per quella domanda). Infine, la piattaforma mette a disposizione dei premi che sono assegnati agli utenti che accumulano una certa quantità di punti bonus. Ogni premio dispone di un nome, una descrizione, una foto, un numero minimo di punti necessari per acquisirlo, ed è inserito da un utente amministratore. Per ogni utente, si vuole tenere traccia dello storico degli eventuali premi vinti.

Infine, si vuole tenere traccia di tutti gli eventi che occorrono nella piattaforma, relativamente all'inserimento di nuovi dati (es. nuovi utenti, nuove domande, nuovi sondaggi, etc). Tali eventi vanno inseriti, sotto forma di messaggi di testo, all'interno di un log, implementato in un' apposita collezione MongoDB.

1.2) Lista delle operazioni

Operazioni che riguardano tutti gli utenti:

- Autenticazione/registrazione sulla piattaforma
- Collegamento ad un dominio di interesse
- Visualizzazione degli inviti a partecipare ad un sondaggio
- Accettazione/rifiuto di un invito Inserimento delle risposte per un sondaggio
- Visualizzazione dei sondaggi cui si è partecipato e delle risposte relative
- Visualizzazione dei premi conseguiti

Operazioni che riguardano SOLO gli utenti PREMIUM:

- Inserimento di una nuova domanda

- Creazione di nuovo sondaggio
- Creazione di un invito ad un sondaggio verso un utente della piattaforma
- Visualizzazione delle risposte di un sondaggio
- Statistiche aggregate su ogni sondaggio (numero di risposte per ogni domanda; per le domande chiuse, distribuzione delle risposte sulle varie opzioni; per le domande aperte, valore medio/minimo e massimo del numero di caratteri)

Operazioni che riguardano SOLO gli utenti AMMINISTRATORI:

- Inserimento di un nuovo premio
- Inserimento di un nuovo dominio

Operazioni che riguardano SOLO le AZIENDE:

- Autenticazione/registrazione sulla piattaforma
- Inserimento di una nuova domanda
- Creazione di nuovo sondaggio
- Invio (automatico) degli inviti
- Statistiche aggregate su ogni sondaggio (numero di risposte per ogni domanda; per le domande chiuse, distribuzione delle risposte sulle varie opzioni; per le domande aperte, valore medio/minimo e massimo del numero di caratteri)

Statistiche (visibili da tutti gli utenti):

- Visualizzare i premi disponibili
- Visualizzare la classifica degli utenti in base al campo totalebonus

1.3) Tavola media dei volumi

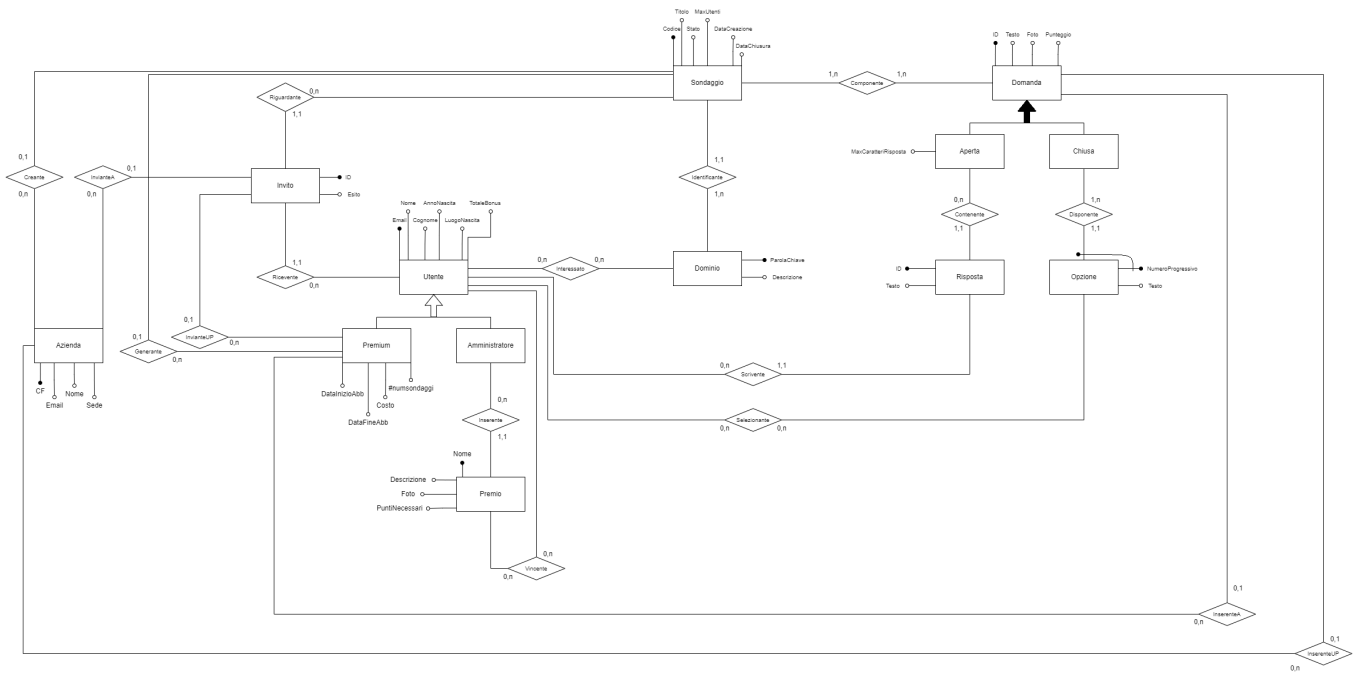
- 5 sondaggi per utente premium
- 20 domande per utente premium
- Si ipotizzano 20 utenti premium

1.4) Glossario dei dati

Non ci sono termini tecnici che richiedano una spiegazione.

2) Progettazione concettuale

2.1) Diagramma E-R



2.2) Dizionario delle entità

Entità	Descrizione	Attributi	Identificatore
Sondaggio	Sondaggio presente in EFORM	Codice, Titolo, Stato, Maxutenti, DataCreazione, DataChiusura	Codice
Dominio	Ambiente di riferimento/campo di esistenza di un sondaggio	ParolaChiave, Descrizione	Parolachiave
Domanda	Domanda che, singolarmente o insieme ad altre compone un sondaggio	ID, Testo, Foto, Punteggio	ID
Aperta	Domanda aperta, che prevede una risposta testuale scritta dall'utente	MaxCaratteriRisposta	ID
Chiusa	Domanda chiusa, che prevede una serie di risposte che l'utente deve selezionare		ID
Opzione	Un'opzione legata ad una specifica domanda chiusa, selezionabile dall'utente	NumeroProgressivo, Testo	NumeroProgressivo, Chiusa
Risposta	Risposta aperta	ID, Testo	ID
Azienda	Un'azienda che può creare sondaggi	CF, Email, Nome, Sede	CF
Invito	Invito relativo ad un sondaggio inviato ad un utente	ID, Esito	ID
Utente	Utente iscritto alla piattaforma EFORM	Email, Nome, Cognome, AnnoNascita, LuogoNascita, TotaleBonus	Email
Premium	Utente abbonato che può creare sondaggi	DataInizioAbb, DataFineAbb, Costo, #numsondaggi	Email
Amministratore	Utente amministratore		Email
Premio	Premio inserito dall'amministratore che può essere vinto dagli utenti in base ai punti in loro possesso	Nome, Descrizione, Foto, PuntiNecessari	Nome

2.3) Dizionario delle relazioni

Relazione	Descrizione	Componenti	Attributi
Componente	Lega un sondaggio a tutte le domande di cui è formato	Sondaggio, Domanda	
Identificante	Il sondaggio è identificato da un dominio	Sondaggio, Dominio	
Contenente	Lega una domanda aperta alla sua risposta	Aperta, Risposta	
Disponente	Lega una domanda chiusa alle sue opzioni di risposta	Chiusa, Opzione	
InserenteA	Lega la domanda all'azienda che la inserisce	Domanda, Azienda	
InserenteUP	Lega la domanda all'utente premium che la inserisce	Domanda, Premium	
Scrivente	Un utente scrive una risposta ad una domanda aperta	Utente, Risposta	
Selezionante	Un utente seleziona un'opzione tra le alternative di una domanda chiusa	Utente, Opzione	
Interessato	Un utente è interessato o meno ad uno o più domini	Utente, Dominio	
Inserente	Un utente amministratore inserisce uno o più premi	Amministratore, Premio	
Vincente	Un utente può vincere un premio in base al suo punteggio	Utente, Premio	
Riguardante	Lega un invito al sondaggio a cui fa riferimento	Invito, Sondaggio	
Ricevente	Lega un utente all'invito che riceve	Utente, Invito	
Creante	Un'azienda può creare dei sondaggi	Azienda, Sondaggio	
Generante	Un utente premium può generare dei sondaggi	Premium, Sondaggio	
InvianteA	Un'azienda può inviare inviti a sondaggi	Azienda, Invito	
InvianteUP	Un utente premium può inviare inviti a sondaggi	Premium, Invito	

2.4) Tavola delle business rules

Regole di vincolo
1) Sondaggio.Stato è un campo ENUM con valori APERTO o CHIUSO
2) Domanda.ID è un intero autoincrementale
3) Invito.Esito può essere ACCETTATO o RIFIUTATO
4) I sondaggi creati dall'azienda devono essere gestiti in forma anonima
5) Utente accetta invito, totaleBonus incrementato di 0,5
6) Utente non può essere contemporaneamente UtentePremium e UtenteAmministratore
7) Ogni occorrenza di Domanda deve appartenere a DomandaAperta o DomandaChiusa
8) Le domande di un sondaggio devono avere lo stesso autore del sondaggio stesso
9) Un invito può essere fatto solo da un'azienda o da un utente premium
10) Un'azienda/Utente Premium può invitare solo per sondaggi da lei/lui creati

3.2.1) Costo operazioni con ridondanza:

- Aggiungere una nuova domanda ad un sondaggio esistente (3 volte/mese, interattiva)

$$3 * 1 * (2 * 2 + 0) = 3 * 4 = 12$$

Faccio una scrittura in Domanda, una scrittura in Componente

- Rimuovere tutte le domande inserite da un utente premium (1 volte/mese, batch)

$$1 * 0.5 * (2 * 40 + 0) = 0.5 * 80 = 40$$

Rimuovo 20 righe in Inserente, rimuovo 20 righe in Domanda

- Rimuovere un sondaggio creato da un utente premium (1 volte/mese, batch)

$$1 * 0.5(2 * 3 + 0) = 0.5 * 6 = 3$$

Rimuovo una riga in generante, rimuovo una riga in Sondaggio, aggiorno #numsondaggi

- Contare il numero di sondaggi associati ad un utente premium (4 volte/mese, interattiva)

$$4 * 1 * (2 * 0 + 1) = 4 * 1 = 4$$

Leggo #numsondaggi

Costo totale con ridondanza

$$12 + 40 + 3 + 4 = 59$$

3.2.2) Costo operazioni senza ridondanza

- Aggiungere una nuova domanda ad un sondaggio esistente (3 volte/mese, interattiva)

$$3 * 1 * (2 * 2 + 0) = 3 * 4 = 12$$

Faccio una scrittura in Domanda, una scrittura in Componente

- Rimuovere tutte le domande inserite da un utente premium (1 volte/mese, batch)

$$1 * 0.5 * (2 * 40 + 0) = 0.5 * 80 = 40$$

Rimuovo 20 righe in Inserente, rimuovo 20 righe in Domanda

- Rimuovere un sondaggio creato da un utente premium (1 volte/mese, batch)

$$1 * 0.5(2 * 2 + 0) = 0.5 * 6 = 2$$

Rimuovo una riga in generante, rimuovo una riga in Sondaggio

- Contare il numero di sondaggi associati ad un utente premium (4 volte/mese, interattiva)

$$4 * 1 * (2 * 0 + 5) = 4 * 5 = 20$$

Leggo 5 righe in Generante

Costo totale senza ridondanza

$$12 + 40 + 3 + 20 = 75$$

3.2.3) Analisi costi memoria

Speedup

$$\frac{\text{CostoSenzaRidondanza}}{\text{CostoConRidondanza}} = \frac{75}{59} \approx 1.27$$

La ridondanza abbatte il costo, è utile.

Memoria

Il campo aggiuntivo #numsondaggi si trova nella tabella UtentePremium. Poniamo, per ipotesi, di avere 20 Utenti Premium.

Il peso del campo, essendo intero, è di 4 Byte.

In tutto occupo:

$$4B * 20 = 80Byte$$

E' utile notare che anche con un volume abbastanza elevato, il costo in memoria risulterebbe comunque irrilevante. Es. per 1000 utenti premium il campo avrebbe un peso di soli 4KB.

Possiamo concludere che mantenere l'attributo ridondante migliora le performance del database.

3.3) Schema Logico

Lista delle tabelle con vincoli di chiavi

- *Sondaggio*(Codice, Titolo, Stato, MaxUtenti, DataCreazione, DataChiusura, ParolachiaveDominio, CFAziendacreante, EmailUtentecreante)
- *ComponenteSondaggioDomanda*(CodiceSondaggio, IDDomanda)
- *Domanda*(ID, Testo, Foto, Punteggio, ApertaChiusa, CFAziendainserente, EmailUtenteinserente)
- *DomandaAperta*(ID, MaxCaratteriRisposta)
- *DomandaChiusa*(ID)
- *Risposta*(ID, Testo, IDDomandaaperta, EmailUtente)
- *Opzione*(Numeroprogressivo, IDDomandachiusa, Testo)
- *Dominio*(Parolachiave, Descrizione)
- *Interessato*(EmailUtente, ParolachiaveDominio)
- *Invito*(ID, Esito, EmailUtente, CodiceSondaggio, CFAziendainvitante, EmailUtenteinvitante)
- *Utente*(Email, Nome, Cognome, Annonascita, Luogonascita, Totalebonus, PAS)
- *SelezionanteUtenteOpzione*(EmailUtente, NumeroprogressivoOpzione, IDDomandachiusaOpzione)
- *UtenteAmministratore*(Email)
- *UtentePremium*(Email, Datainizioabbonamento, Datafineabbonamento, Costo, #numsondaggi)
- *Premio*(Nome, Descrizione, Foto, Puntinecessari, EmailUtenteAmministratore)
- *Vincente*(NomePremio, EmailUtente)
- *Azienda*(CE, Email, Nome, Sede)

Lista dei vincoli inter-relazionali

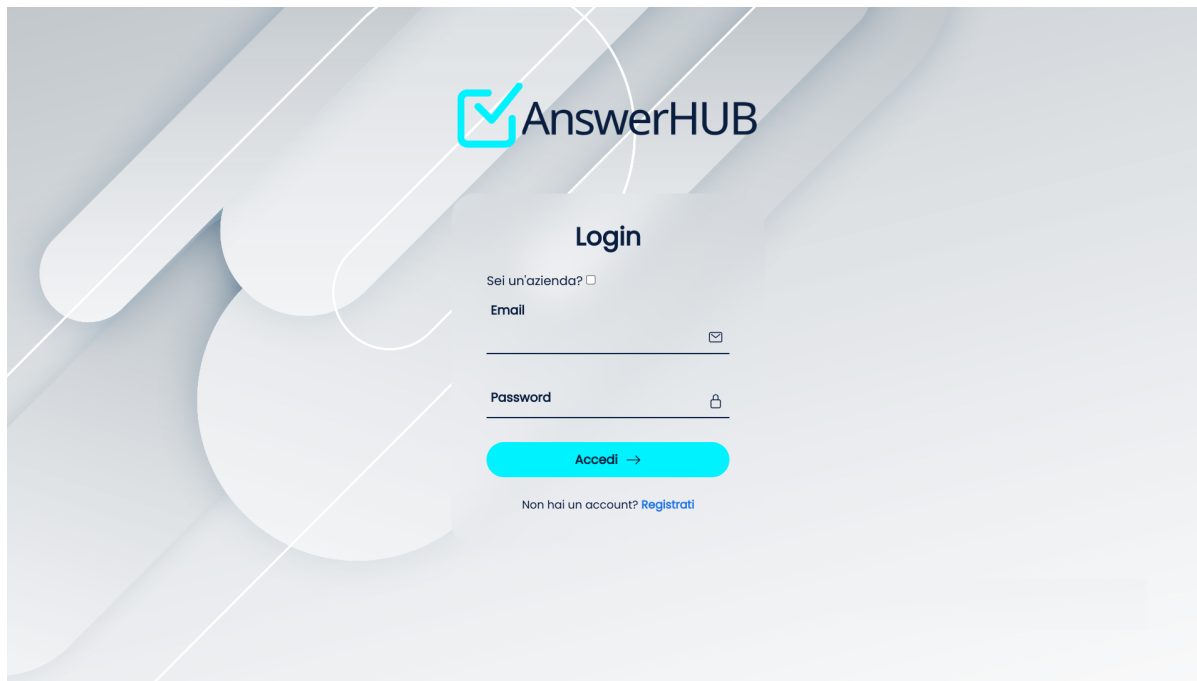
- Componente.CodiceSondaggio → Sondaggio.Codice
Componente.IDDomanda → Domanda.ID
- DomandaAperta.IDDomanda → Domanda.ID
- DomandaChiusa.IDDomanda → Domanda.ID
- Risposta.IDDomandaaperta → DomandaAperta.IDDomanda
Risposta.EmailUtente → Utente.Email
- Opzione.IDDomandachiusa → DomandaChiusa.IDDomanda

- Sondaggio.ParolachiaveDominio → Dominio.Parolachiave
Sondaggio.CFAziendacreante → Azienda.CF
Sondaggio.EmailUtentecreante → UtentePremium.Email
- Interessato.EmailUtente → Utente.Email
Interessato.ParolachiaveDominio → Dominio.Parolachiave
- Invito.EmailUtente → Utente.Email
Invito.CodiceSondaggio → Sondaggio.Codice
Invito.CFAziendainvitante → Azienda.CF
Invito.EmailUtenteinvitante → Utente.Email
- SelezionanteUtenteOpzione.EmailUtente → Utente.Email
SelezionanteUtenteOpzionemy.NumeroprogressivoOpzione,
Selezionante.IDDomandachiusaOpzione → Opzione.Numeroprogressivo,
Opzione.IDDomandachiusa
- UtenteAmministratore.Email → Utente.Email
- UtentePremium.Email → Utente.Email
- Premio.EmailUtenteAmministratore → UtenteAmministratore.Email
- Vincente.NomePremio → Premio.Nome
Vincente.EmailUtente → Utente.Email
- Domanda.CFAziendainserente → Azienda.CF
Domanda.EmailUtenteinserente → UtentePremium.Email

Le relazioni (0,1)-(0,n) sono state tradotte accorrandole nelle entità con cardinalità (0,1).
In questo modo si evita la complessità di dover gestire ulteriori 4 tabelle.

4) Descrizione -ad alto livello- delle funzionalità dell'applicazione Web

Login e accesso



Una volta collegati all'applicazione web AnswerHUB, viene presentata la pagina di login. Qui viene mostrato l'accesso come utente ma si può selezionare una checkbox per effettuare l'accesso come azienda:

- l'accesso come utente viene gestito con email e password;
- l'accesso come azienda viene gestito tramite il suo codice fiscale.

Una volta eseguito l'accesso come utente:

- se l'utente è un utente semplice, verrà mostrata una dashboard con le funzionalità "semplici", cioè quelle dedicate a tutti gli utenti
- se l'utente è un utente premium, verrà mostrata una dashboard con le funzionalità "premium", inoltre grazie ad un link vi è la possibilità di passare alla dashboard delle funzionalità "semplici"
- se l'utente è un utente amministratore, verrà mostrata una dashboard con le funzionalità "amministratore", inoltre grazie ad un link vi è la possibilità di passare alla dashboard delle funzionalità "semplici"

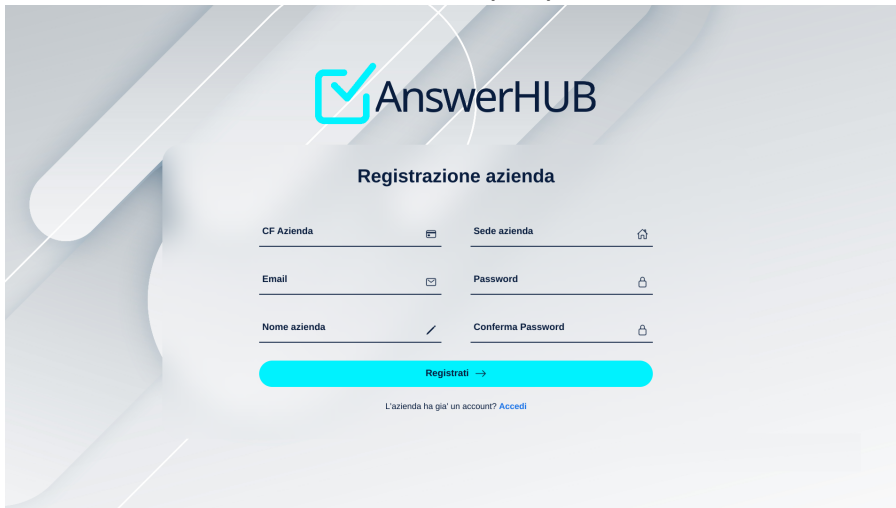
Si è optato per questa soluzione in quanto gli utenti premium e amministratori sono comunque utenti della piattaforma dunque gli deve essere consentito effettuare tutte le operazioni dell'utente "semplice" (che non è né premium, né amministratore).

Registrazione

Dalla pagina di login si può passare, attraverso un apposito collegamento, alla pagina di registrazione.

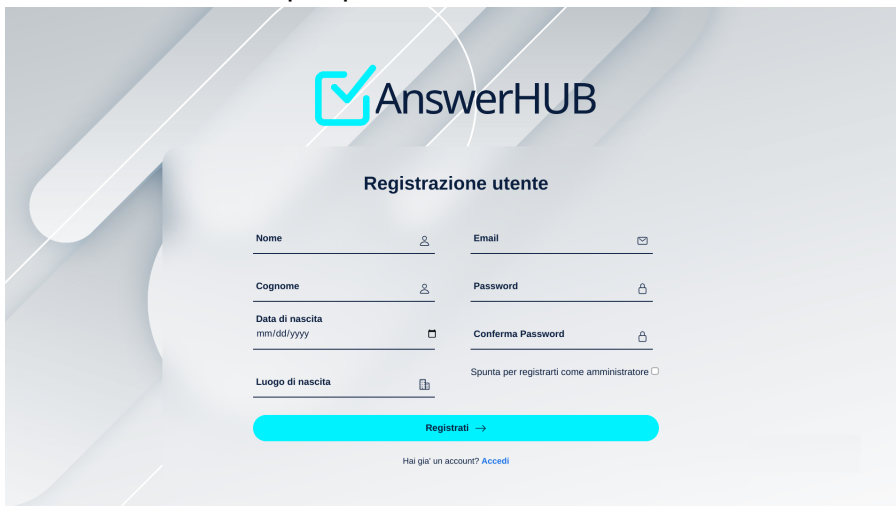
Se nella pagina di login è stata selezionata la checkbox riguardo l'accesso come azienda,

allora il collegamento alla registrazione rimanderà ad un form per la registrazione come azienda (che richiede tutti i campi tipici di un'azienda):



The screenshot shows the 'Registrazione azienda' (Company Registration) form on the AnswerHUB website. The form is titled 'Registrazione azienda' and features the AnswerHUB logo at the top. It contains the following fields: 'CF Azienda' (Company Tax Code), 'Sede azienda' (Company Address), 'Email', 'Password', 'Nome azienda' (Company Name), and 'Conferma Password' (Confirm Password). A red 'Registrati' button with a right arrow is at the bottom. Below the button, there is a link: 'L'azienda ha già un account? [Accedi](#)'.

altrimenti il collegamento rimanderà ad un form per la registrazione come utente (che richiede tutti i campi tipici di un utente):



The screenshot shows the 'Registrazione utente' (User Registration) form on the AnswerHUB website. The form is titled 'Registrazione utente' and features the AnswerHUB logo at the top. It contains the following fields: 'Nome' (First Name), 'Cognome' (Last Name), 'Data di nascita' (Date of Birth), 'Luogo di nascita' (Place of Birth), 'Email', 'Password', and 'Conferma Password'. There is also a checkbox labeled 'Spunta per registrarti come amministratore' (Check to register as administrator). A red 'Registrati' button with a right arrow is at the bottom. Below the button, there is a link: 'Hai già un account? [Accedi](#)'.

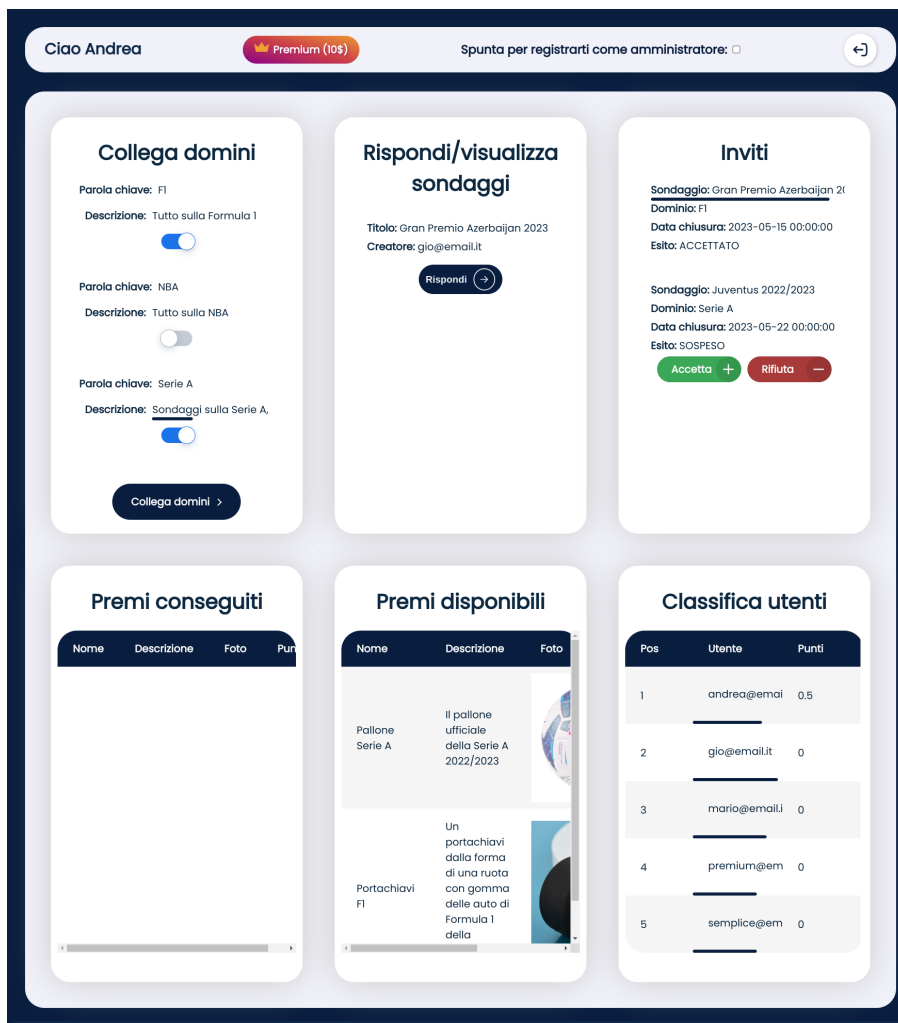
Qui l'utente può registrarsi come amministratore: spuntando l'apposita checkbox, verrà mostrato un campo che richiede il codice 66.

La scelta del numero non ha un particolare significato, si è optato per questa soluzione in quanto ci si è immaginato che l'utente può contattare tramite mail l'azienda AnswerHUB e richiedere il codice per diventare amministratore.

La possibilità di diventare amministratore viene data anche se ci si registra, in prima istanza, come semplici utenti.

Tale possibilità non viene data se l'utente è premium in quanto, come da specifiche, un utente può essere premium oppure amministratore.

Utente semplice



Una volta effettuato l'accesso, se l'utente non è né premium né amministratore, verrà mostrata la dashboard "semplice" che consiste in degli "spazi" ognuno dei quali dedicato ad ogni funzionalità.

Qui l'utente può:

- **Diventare premium:** cliccando sull'apposito bottone, l'utente può attivare un abbonamento annuale e diventare premium
- **Diventare amministratore:** spuntando l'apposita casella, verrà mostrato un campo di input in cui inserire il codice 66.
- **Collega domini:** l'utente può mostrare o rimuovere l'interesse per un dominio. Se l'utente rimuove l'interesse ad un dominio, da questo momento in poi non riceverà più inviti a sondaggi di quel dominio. I sondaggi accettati o in sospeso fino a quel momento vengono però mantenuti in quanto al momento dell'invito, l'utente era interessato a quel dominio.
- **Rispondi/Visualizza sondaggi:** Da qui l'utente può rispondere ai sondaggi il cui invito è stato accettato. Una volta risposto, l'utente potrà visualizzare le risposte date (il bottone "Rispondi" diventa "Visualizza")

Rispondi: Gran Premio Azerbaijan 2021

Chi ha vinto?

Punteggio: 1


Seleziona l'opzione:

☐ Max Verstappen

☒ Sergio Perez

☐ Charles Leclerc

Chi è partito dalla Pole Position



Punteggio: 1

Seleziona l'opzione:

☐ George Russell

☒ Charles Leclerc

☐ Max Verstappen

☐ Nessuno di questi

Invia risposte >

Visualizza risposte: Gran Premio Azerb

Chi ha vinto?

Punteggio: 1


Opzione selezionata:

☒ Sergio Perez

☐ Max Verstappen

☐ Charles Leclerc

Chi è partito dalla Pole Position



Punteggio: 1

Opzione selezionata:

☒ Charles Leclerc

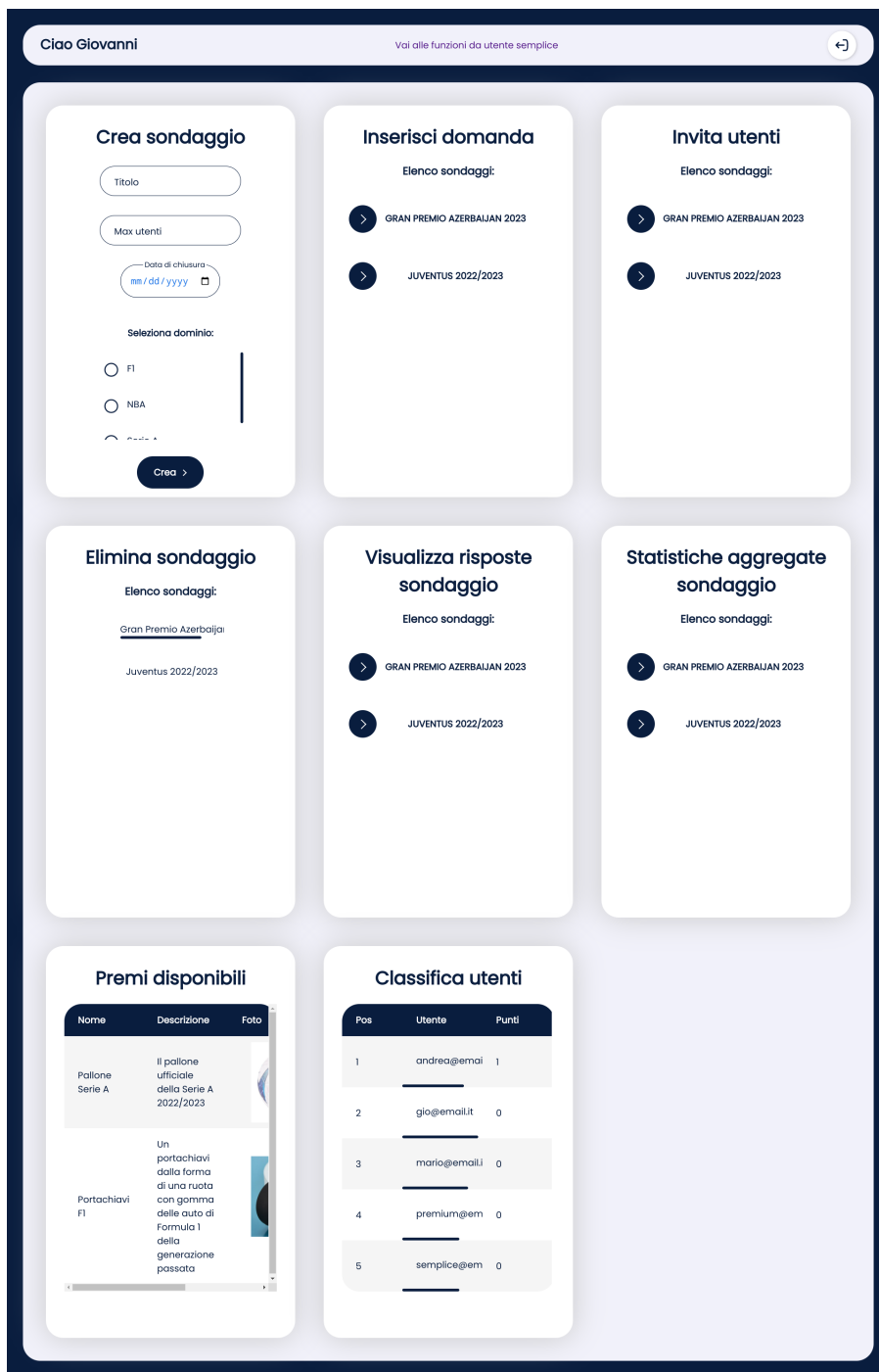
☐ George Russell

☐ Max Verstappen

☐ Nessuno di questi

- **Inviti:** da qui l'utente può visualizzare tutti gli inviti a partecipare ad un sondaggio che ha ricevuto, può accettare/rifiutare gli inviti in sospeso e vedere quelli che ha accettato o rifiutato grazie all'apposito campo "Esito".
- **Premi conseguiti:** l'utente visualizza i premi conseguiti nel momento in cui il suo totale bonus è almeno pari al punteggio richiesto per il premio.
E' stata gestita la casistica in cui un amministratore inserisce un premio con punteggio inferiore al totale bonus disponibile dell'utente, in questo caso il premio viene assegnato automaticamente.
- **Premi disponibili:** viene mostrata una tabella (navigabile orizzontalmente e verticalmente) che mostra nome, descrizione, foto e punti necessari per ogni premio
- **Classifica utenti:** viene mostrata una tabella che mostra posizione, nome utente e punti di tutti gli utenti registrati sulla piattaforma.

Utente Premium



Una volta effettuato l'accesso, se l'utente è premium, gli verrà mostrata la dashboard con le funzionalità premium.

Da qui l'utente può:

- **Vai alle funzioni da utente semplice:** cliccando, l'utente viene rimandato ad una dashboard identica a quella dell'utente semplice, da qui potrà svolgere tutte le operazioni viste in precedenza.

In questo caso, inoltre, viene mostrato un collegamento per tornare alla dashboard premium.

- **Crea sondaggio:** l'utente inserisce i dati utili alla creazione di un sondaggio, seleziona il dominio dall'apposita lista e clicca sul bottone per creare il sondaggio.
- **Inserisci domanda:** una volta creato, il sondaggio verrà mostrato in questa sezione. Cliccando sul sondaggio che gli interessa, l'utente verrà portato ad una nuova pagina in cui si potranno creare le domande.

Nella nuova pagina l'utente può creare una domanda, se vuole creare una domanda aperta spunta l'apposita checkbox (successivamente verrà mostrato un campo per indicare il massimo numero di caratteri consentiti alla risposta aperta), altrimenti la domanda sarà chiusa.

Sulla destra verrà mostrato l'elenco di domande; l'utente può cliccare sulla domanda chiusa e verrà reindirizzato ad un'altra pagina per gestire le opzioni (immagine di seguito).

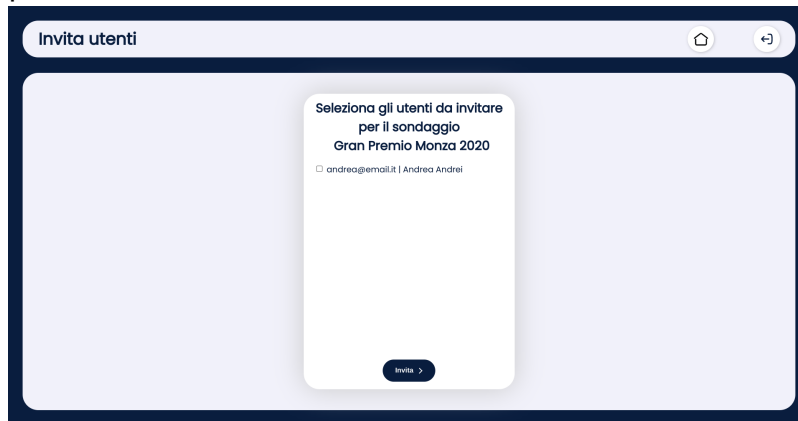
Fin quando nessun utente invitato ha accettato l'invito al sondaggio, l'utente può eliminare il sondaggio stesso, eliminare delle domande del sondaggio, eliminare delle opzioni di una domanda chiusa del sondaggio.

Nel momento in cui almeno un utente invitato accetta l'invito, le funzionalità di eliminazione non sono più disponibili.

Si è optato per questa soluzione per mantenere consistenti le domande del sondaggio tra gli utenti che hanno già accettato l'invito e potenzialmente già risposto al sondaggio, e nuovi utenti che ancora non hanno accettato e dunque risposto.

- **Invita:** l'utente non può invitare nessuno se il sondaggio non ha domande. Una volta che l'utente ha inserito almeno una domanda e tutte le domande chiuse, se ci sono, hanno almeno un'opzione; l'utente può inviare gli inviti.

In tal caso viene mostrato l'elenco degli utenti interessati al dominio del sondaggio, se ci sono, altrimenti verrà notificato che non ci sono utenti interessati e l'utente non potrà invitare nessuno.

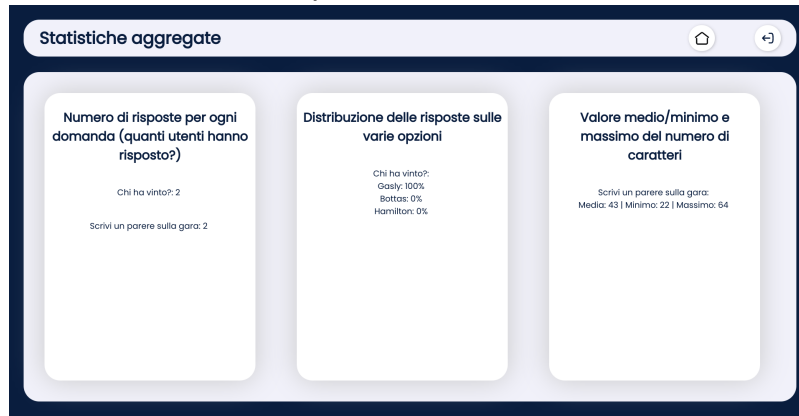


- **Elimina sondaggio:** viene mostrata la lista dei sondaggi e, come detto in precedenza, se nessun utente ha ancora accettato l'invito al sondaggio, questo può essere eliminato.
- **Visualizza risposte sondaggio:** viene mostrata la lista dei sondaggi, cliccando su un sondaggio vengono visualizzate le domande con le risposte degli utenti che hanno risposto al sondaggio.



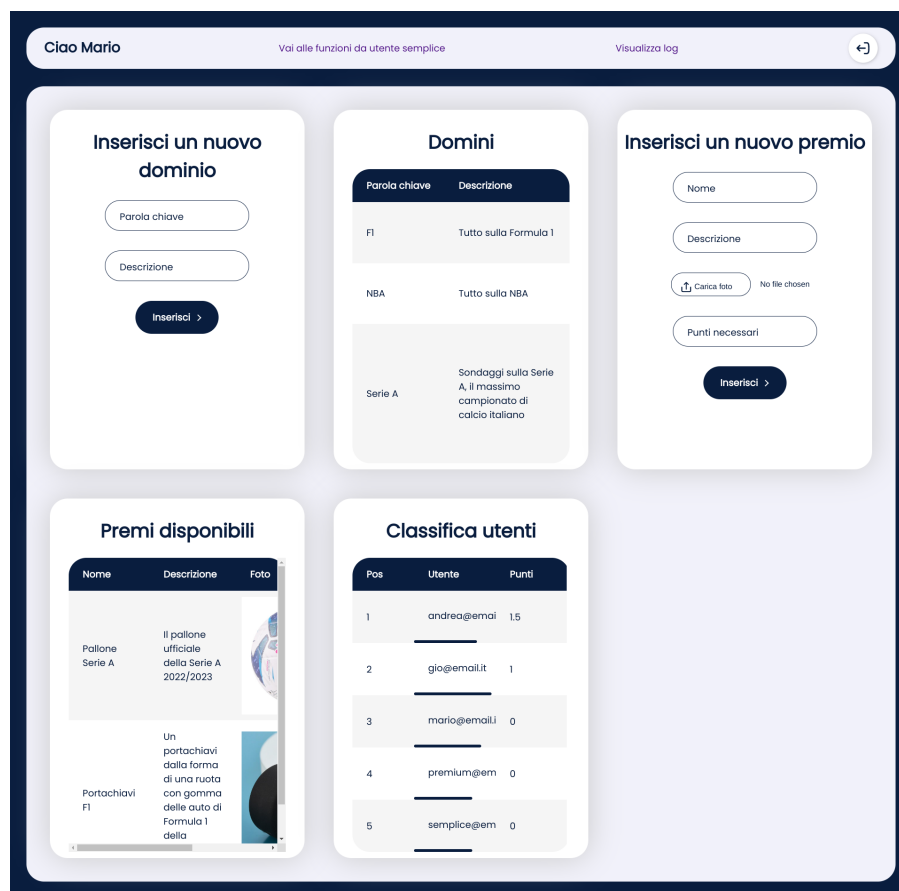
- **Statistiche aggregate sondaggio:** viene mostrata la lista dei sondaggio, cliccando su un sondaggio vengono visualizzate le statistiche aggregate (numero risposte,

distribuzione sulle opzioni, valore medio/minimo e massimo del numero di caratteri)



- **Premi disponibili**: viene mostrata una tabella con tutti i premi disponibili
- **Classifica utenti**: viene mostrata la classifica degli utenti

Utente Amministratore

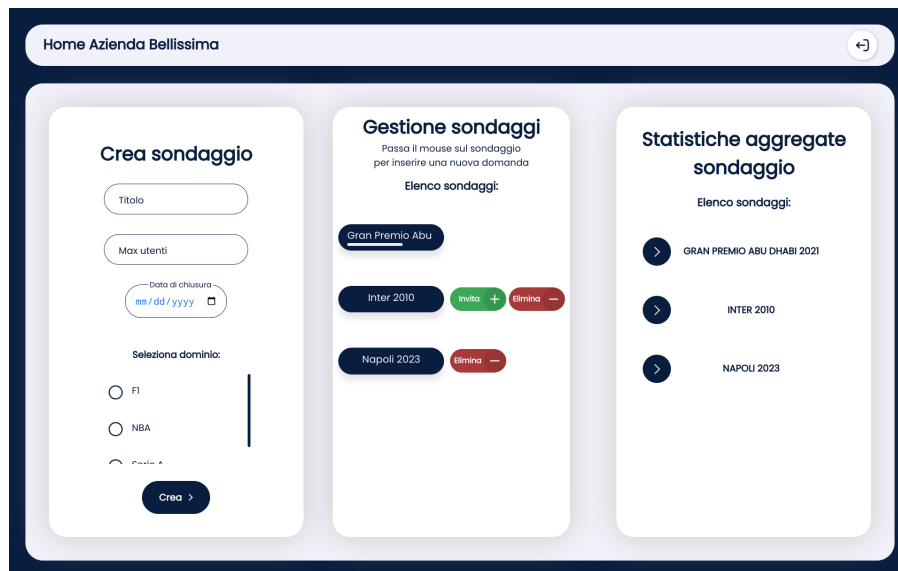


Una volta effettuato l'accesso, se l'utente è amministratore, gli verrà mostrata la dashboard con le funzionalità da amministratore.

Da qui l'utente può:

- **Vai alle funzioni da utente semplice:** cliccando, l'utente viene rimandato ad una dashboard identica a quella dell'utente semplice, da qui potrà svolgere tutte le operazioni viste in precedente.
In questo caso, inoltre, viene mostrato un collegamento per tornare alla dashboard amministratore.
- **Visualizza log:** l'utente amministratore può visualizzare il log delle operazioni effettuate sul database (implementato, come da specifiche, con mongodb).
- **Inserisci un nuovo dominio:** l'utente può indicare parola chiave e descrizione e inserire un nuovo dominio
- **Domini:** viene mostrata una tabella con tutti i domini specificandone parola chiave e descrizione
- **Inserisci un nuovo premio:** un form per inserire un nuovo premio specificando nome, descrizione, obbligo di caricare un'immagine come da specifiche, punti necessari al conseguimento del premio.
Come specificato in precedenza, se l'utente amministratore inserisce un premio e uno o più utenti hanno un totale bonus superiore, il premio viene automaticamente assegnato a questo utenti.
- **Premi disponibili:** viene mostrata una tabella con tutti i premi disponibili
- **Classifica utenti:** viene mostrata la classifica degli utenti

Azienda



Una volta effettuato l'accesso, l'azienda vede una dashboard con le sue funzionalità:

- **Crea sondaggio:** stesso funzionamento di premium
- **Gestione sondaggi:** qui l'azienda vede la lista di sondaggi da essa creati, cliccando sul nome del sondaggio può andarne a gestire le domande e le opzioni per le domande chiuse (verrà reindirizzata ad una pagina per la gestione delle domande da

cui potrà essere reindirizzata ad una pagina per la gestione delle opzioni, stesso funzionamento di premium).

I sondaggi, le domande e le opzioni possono essere eliminate fin quando nessun utente ha accettato l'invito al sondaggio.

Nel caso dell'esempio di cui sopra, per il sondaggio "Gran Premio Abu Dhabi 2021" un utente ha accettato l'invito dunque non è più possibile eliminarlo, per il sondaggio "Inter 2010" non sono ancora stati mandati gli inviti, per il sondaggio "Napoli 2023" gli inviti sono stati mandati ma nessuno ha ancora accettato, dunque è ancora possibile eliminare il sondaggio

Nel caso dell'azienda l'invito è automatico, si è deciso di invitare gli utenti in base all'ordine in cui questi ultimi si sono mostrati interessati al dominio di appartenenza del sondaggio (in una logica di fairness). Un utente che rimuove il dominio ed in un secondo momento lo reinserisce, perde il suo "diritto di prelazione" in quanto scivolerà in fondo alla lista di utenti interessati.

Dovendo specificare in fase di creazione del sondaggio un numero massimo di utenti che possono partecipare, dunque, verranno presi i primi x utenti che hanno mostrato interesse al dominio.

- **Statistiche aggregate sondaggio:** stesso funzionamento di premium

Codice SQL completo dello schema della base di dati

```
# Utile per gestire le foreign keys (update/delete)
set foreign_key_checks = 1;

# Utile per poter usare nel where una condizione che confronta campi non
primary key
SET SQL_SAFE_UPDATES=0;

CREATE DATABASE IF NOT EXISTS AnswerHUB_DB;

# rimuove db pre-esistente con stesso nome
DROP DATABASE IF EXISTS AnswerHUB_DB;

# creazione database
CREATE DATABASE IF NOT EXISTS AnswerHUB_DB;

# seleziona database
USE AnswerHUB_DB;
```

```
# creazione tabelle
CREATE TABLE IF NOT EXISTS controllo_evento_log (
Data timestamp,
Messaggio varchar(255),
PRIMARY KEY (Data, Messaggio)
)
engine = innodb;

CREATE TABLE IF NOT EXISTS Utente (
Email varchar(255),
Pwd varchar(30),
Nome varchar(30),
Cognome varchar(30),
Annonascita timestamp,
Luogonascita varchar(50),
Totalebonus float,
PAS ENUM ('PREMIUM', 'AMMINISTRATORE', 'SEMPLICE'),
PRIMARY KEY (Email)
)
engine = innodb;

CREATE TABLE IF NOT EXISTS UtenteAmministratore (
Email varchar(255),
FOREIGN KEY (Email) REFERENCES Utente(Email) ON DELETE cascade ON UPDATE
cascade,
PRIMARY KEY (Email)
)
engine = innodb;

CREATE TABLE IF NOT EXISTS UtentePremium (
Email varchar(255),
Datainizioabbonamento timestamp,
Datafineabbonamento timestamp,
Costo decimal,
numsondaggi integer DEFAULT 0,
FOREIGN KEY (Email) REFERENCES Utente(Email) ON DELETE cascade ON UPDATE
cascade,
PRIMARY KEY (Email)
)
```

```
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Azienda (  
CF varchar(16), -- il CF di un'azienda e' di 16 caratteri  
Pwd varchar(30),  
Email varchar(255) UNIQUE,  
Nome text,  
Sede text,  
PRIMARY KEY (CF)  
)  
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Dominio (  
Parolachiave varchar(255),  
Descrizione text,  
PRIMARY KEY (Parolachiave)  
)  
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Sondaggio (  
Codice integer AUTO_INCREMENT, #aggiunto  
Titolo varchar(255),  
Stato ENUM ('APERTO', 'CHIUSO'),  
MaxUtenti integer,  
DataCreazione timestamp, -- formato AAAA-MM-GG  
DataChiusura timestamp,  
ParolachiaveDominio varchar(255),  
CFAziendacreante varchar(16),  
EmailUtentecreante varchar(255),  
FOREIGN KEY (ParolachiaveDominio) REFERENCES Dominio(Parolachiave) ON DELETE  
cascade ON UPDATE cascade,  
FOREIGN KEY (CFAziendacreante) REFERENCES Azienda(CF) ON DELETE cascade ON  
UPDATE cascade,  
FOREIGN KEY (EmailUtentecreante) REFERENCES UtentePremium(Email) ON DELETE  
cascade ON UPDATE cascade,  
PRIMARY KEY (Codice)  
)  
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Domanda (  

```

```

ID integer auto_increment,
Testo varchar(3000),
Foto longblob, -- file fino a 4GB
Punteggio integer,
ApertaChiusa ENUM ('APERTA', 'CHIUSA'), -- aperta -> true = 1, chiusa ->
false = 0
CFAziendainserente varchar(16),
EmailUtenteinserente varchar(255),
FOREIGN KEY (CFAziendainserente) REFERENCES Azienda(CF) ON DELETE cascade ON
UPDATE cascade,
FOREIGN KEY (EmailUtenteinserente) REFERENCES UtentePremium(Email) ON DELETE
cascade ON UPDATE cascade,
PRIMARY KEY (ID)
)
engine = innodb;

```

```

CREATE TABLE IF NOT EXISTS ComponenteSondaggioDomanda (
CodiceSondaggio integer,
IDDomanda integer,
PRIMARY KEY (CodiceSondaggio, IDDomanda),
FOREIGN KEY (CodiceSondaggio) REFERENCES Sondaggio(Codice) ON DELETE cascade
ON UPDATE cascade,
FOREIGN KEY (IDDomanda) REFERENCES Domanda(ID) ON DELETE cascade ON UPDATE
cascade
)
engine = innodb;

```

```

CREATE TABLE IF NOT EXISTS DomandaAperta (
ID integer,
MaxCaratteriRisposta integer,
FOREIGN KEY (ID) REFERENCES Domanda(ID) ON DELETE cascade ON UPDATE cascade,
PRIMARY KEY (ID)
)
engine = innodb;

```

```

CREATE TABLE IF NOT EXISTS DomandaChiusa (
ID integer,
FOREIGN KEY (ID) REFERENCES Domanda(ID) ON DELETE cascade ON UPDATE cascade,
PRIMARY KEY (ID)
)

```

```
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Risposta (  
  ID integer AUTO_INCREMENT,  
  Testo text,  
  IDDomandaaperta integer,  
  EmailUtente varchar(255),  
  FOREIGN KEY (IDDomandaaperta) REFERENCES DomandaAperta(ID) ON DELETE cascade  
  ON UPDATE cascade,  
  FOREIGN KEY (EmailUtente) REFERENCES Utente(Email) ON DELETE cascade ON  
  UPDATE cascade,  
  PRIMARY KEY (ID)  
)  
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Opzione (  
  IDDomandachiusa integer NOT NULL,  
  Numeroprogressivo integer NOT NULL, -- un trigger lo rende auto_increment  
  rispetto a IDDomandachiusa  
  Testo text,  
  FOREIGN KEY (IDDomandachiusa) REFERENCES DomandaChiusa(ID) ON DELETE cascade  
  ON UPDATE cascade,  
  PRIMARY KEY (IDDomandachiusa, Numeroprogressivo)  
)  
engine = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS Interessato (  
  EmailUtente varchar(255),  
  ParolachiaveDominio varchar(255),  
  FOREIGN KEY (EmailUtente) REFERENCES Utente(Email) ON DELETE cascade ON  
  UPDATE cascade,  
  FOREIGN KEY (ParolachiaveDominio) REFERENCES Dominio(Parolachiave) ON DELETE  
  cascade ON UPDATE cascade,  
  PRIMARY KEY (EmailUtente, ParolachiaveDominio)  
)  
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Invito (  
  ID integer AUTO_INCREMENT, #aggiunto  
  Esito ENUM ('SOSPESO', 'ACCETTATO', 'RIFIUTATO'),
```



```
EmailUtente varchar(255),
CodiceSondaggio integer,
CFAziendainvitante varchar(16),
EmailUtenteinvitante varchar(255),
FOREIGN KEY (EmailUtente) REFERENCES Utente(Email) ON DELETE cascade ON
UPDATE cascade,
FOREIGN KEY (CodiceSondaggio) REFERENCES Sondaggio(Codice) ON DELETE cascade
ON UPDATE cascade,
FOREIGN KEY (CFAziendainvitante) REFERENCES Azienda(CF) ON DELETE cascade ON
UPDATE cascade,
FOREIGN KEY (EmailUtenteinvitante) REFERENCES UtentePremium(Email) ON DELETE
cascade ON UPDATE cascade,
PRIMARY KEY (ID)
)
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS SelezionanteUtenteOpzione(
EmailUtente varchar(255),
IDDomandachiusaOpzione integer,
NumeroprogressivoOpzione integer,
FOREIGN KEY (EmailUtente) REFERENCES Utente(Email) ON DELETE cascade ON
UPDATE cascade,
FOREIGN KEY (IDDomandachiusaOpzione, NumeroprogressivoOpzione) REFERENCES
Opzione(IDDomandachiusa, Numeroprogressivo) ON DELETE cascade ON UPDATE
cascade,
PRIMARY KEY (EmailUtente, NumeroprogressivoOpzione, IDDomandachiusaOpzione)
)
engine = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS Premio (
Nome varchar(255),
Descrizione text,
Foto longblob,
Puntinecessari float,
EmailUtenteAmministratore varchar(255),
FOREIGN KEY (EmailUtenteAmministratore) REFERENCES
UtenteAmministratore(Email) ON DELETE cascade ON UPDATE cascade,
PRIMARY KEY (Nome)
)
engine = innodb;
```

```
CREATE TABLE IF NOT EXISTS Vincente (  
  NomePremio varchar(255),  
  EmailUtente varchar(255),  
  FOREIGN KEY (NomePremio) REFERENCES Premio(Nome) ON DELETE cascade ON UPDATE  
  cascade,  
  FOREIGN KEY (EmailUtente) REFERENCES Utente(Email) ON DELETE cascade ON  
  UPDATE cascade,  
  PRIMARY KEY (NomePremio, EmailUtente)  
)  
engine = innodb;
```

```
# TRIGGER
```

```
# trigger per rendere auto_increment Opzione.Numeroprogressivo rispetto a  
Opzione.IDDomandachiusa
```

```
DELIMITER //
```

```
CREATE TRIGGER auto_increment_Numeroprogressivo
```

```
BEFORE INSERT ON Opzione
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE nNumprog INT;
```

```
    SELECT COALESCE(MAX(Numeroprogressivo), 0) + 1 -- ritorna il primo  
valore non nullo tra max(numeroprogressivo e 0), e lo incrementa di 1
```

```
    INTO    nNumprog
```

```
    FROM    Opzione
```

```
    WHERE   IDDomandachiusa = NEW.IDDomandachiusa;
```

```
    SET NEW.Numeroprogressivo = nNumprog;
```

```
END
```

```
//
```

```
DELIMITER ;
```

```
# trigger per popolare automaticamente DomandaAperta/DomandaChiusa all  
inserimento di una nuova Domanda sulla base del campo boolean ApertaChiusa
```

```
DELIMITER //
```

```
CREATE TRIGGER PopolaDomandaApertaChiusa
```

```
AFTER INSERT ON Domanda
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF (NEW.ApertaChiusa = 'APERTA') THEN
```

```

        INSERT INTO DomandaAperta(ID) VALUES (NEW.ID);
    END IF;
    IF (NEW.ApertaChiusa = 'CHIUSA') THEN
        INSERT INTO DomandaChiusa(ID) VALUES (NEW.ID);
    END IF;
END
//
DELIMITER ;

# trigger per popolare automaticamente UtentePremium/UtenteAmministratore
all inserimento di un nuovo utente se è Premium/Amministratore
DELIMITER //
CREATE TRIGGER PopolaUtentePremiumAmministratore
AFTER INSERT ON Utente
FOR EACH ROW
BEGIN
    IF (NEW.PAS = 'PREMIUM') THEN
        INSERT INTO UtentePremium(Email, Datainizioabbonamento,
Datafineabbonamento, Costo)
        VALUES (NEW.Email, NOW(), DATE_ADD(DATE_ADD(CURDATE(), INTERVAL 1
DAY), INTERVAL 1 YEAR), 10);
    END IF;
    IF (NEW.PAS = 'AMMINISTRATORE') THEN
        INSERT INTO UtenteAmministratore(Email) VALUES (NEW.Email);
    END IF;
END
//
DELIMITER ;

# trigger per popolare automaticamente UtentePremium/UtenteAmministratore se
un utente semplice diventa in un secondo momento Premium/Amministratore
DELIMITER //
CREATE TRIGGER PopolaUtentePremiumAmministratoreDopoUpdate
AFTER UPDATE ON Utente
FOR EACH ROW
BEGIN
    IF (NEW.PAS = 'PREMIUM' AND OLD.PAS <> 'PREMIUM') THEN -- OLD.PAS messo
per evitare che il trigger si attivi anche quando si attiva il trigger
IncrementaTotaleBonus che fa un update su Invito ma l'utente era già premium
quindi evita di fare una insert duplicata

```

```

        INSERT INTO UtentePremium(Email, Datainizioabbonamento,
Datafineabbonamento, Costo)
        VALUES (NEW.Email, NOW(), DATE_ADD(DATE_ADD(CURDATE(), INTERVAL 1
DAY), INTERVAL 1 YEAR), 10);
    END IF;

    IF (NEW.PAS = 'AMMINISTRATORE' AND OLD.PAS <> 'AMMINISTRATORE') THEN
        INSERT INTO UtenteAmministratore(Email) VALUES (NEW.Email);
    END IF;
END;
//
DELIMITER ;

# trigger per rimuovere automaticamente UtentePremium/UtenteAmministratore
se un utente Premium/Amministratore diventa Semplice in un secondo momento
DELIMITER //
CREATE TRIGGER RimuoviUtentePremiumAmministratore
AFTER UPDATE ON Utente
FOR EACH ROW
BEGIN
    IF (OLD.PAS = 'PREMIUM' AND NEW.PAS = 'SEMPLICE') THEN
        DELETE FROM UtentePremium WHERE (Email = NEW.Email);
    END IF;
    IF (OLD.PAS = 'AMMINISTRATORE' AND NEW.PAS = 'SEMPLICE') THEN
        DELETE FROM UtenteAmministratore WHERE (Email = NEW.Email);
    END IF;
END
//
DELIMITER ;

# EVENT per far tornare semplice un utente premium a cui scade l'abbonamento
# questo evento è complementare al trigger
RimuoviUtentePremiumAmministratore
/*NB: questo comporta una serie di cose, a questo punto cosa succede ai
sondaggi da lui creati quando era premium?
Diventa un utente semplice a tutti gli effetti. Se vuole controllare i
sondaggi già fatti, deve riabbonarsi*/
DELIMITER //
CREATE EVENT ControlloDatafineabbonamento
ON SCHEDULE EVERY 1 DAY STARTS NOW()

```

```

DO
BEGIN
    -- Soluzione usata per evitare errore 1442, stessa tabella usata in una
    subquery

    -- creo una tabella temporanea per salvare gli indirizzi email degli
    utenti premium a cui è scaduta l'iscrizione
    CREATE TEMPORARY TABLE UtentiScaduti AS
    SELECT Email FROM UtentePremium WHERE DATE(Datafineabbonamento) =
    CURDATE();

    -- aggiorno il valore PAS degli utenti la cui mail corrisponde alle
    mail trovate prima (la tabella UtentePremium sarà "pulita" grazie al trigger
    RimuoviUtentePremiumAmministratore)
    UPDATE Utente SET PAS = 'SEMPLICE' WHERE Email IN (SELECT Email FROM
    UtentiScaduti);

    -- elimino la tabella temporanea
    DROP TABLE UtentiScaduti;
END
//
ALTER EVENT ContolloDatafineabbonamento ENABLE;
//
DELIMITER ;

/*
Utilizzare un trigger/evento per implementare l'operazione cambio di stato
un sondaggio.
Un sondaggio diventa CHIUSO quando (i) la data di chiusura è anteriore alla
data attuale
OPPURE (ii) è stato raggiunto un numero di utenti partecipanti (=che hanno
accettato l'invito) pari a maxutenti.
*/
DELIMITER //
CREATE TRIGGER CambioStatoSondaggioMaxUtenti AFTER UPDATE ON Invito
FOR EACH ROW
BEGIN
    IF (NEW.Esito <> OLD.Esito) THEN
        UPDATE Sondaggio SET Stato = 'Chiuso' WHERE MaxUtenti = (SELECT
        COUNT(*) FROM Invito WHERE (Invito.Esito = 1) AND (Invito.CodiceSondaggio =

```

```

Sondaggio.Codice));
    END IF;
END;

CREATE TRIGGER CambioStatoSondaggioData
BEFORE UPDATE ON Sondaggio
FOR EACH ROW
BEGIN
    IF (NEW.DataChiusura <= CURDATE()) THEN
        SET NEW.Stato = 'CHIUSO';
    END IF;
END;
//
DELIMITER ;

/*
Utilizzare un trigger per implementare l'operazione di assegnamento di un
premio ad un utente,
quando il campo totalebonus dell'utente diventa maggiore del numero di punti
richiesti dal premio stesso
*/
DELIMITER //
CREATE TRIGGER AssegnamentoPremio AFTER UPDATE ON Utente
FOR EACH ROW
BEGIN
    DECLARE done BOOLEAN DEFAULT FALSE;
    DECLARE NomePremio_locale varchar(255);
    DECLARE CurPremi CURSOR FOR SELECT Nome FROM Premio WHERE Puntinecessari
<= NEW.Totalebonus;
    -- vengono salvati nel cursore CurPremi i nomi dei premi con il
    -- "Puntinecessari" inferiore al Totalebonus dell'utente
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN CurPremi;

    WHILE NOT done DO
        FETCH CurPremi INTO NomePremio_locale; -- for each
        IF (NomePremio_locale IS NOT NULL AND NOT done) THEN -- se c'e' un
valore e non hai finito, allora inserisci in Vincente
            IF NOT EXISTS (SELECT * FROM Vincente WHERE NomePremio =

```

```

NomePremio_locale AND EmailUtente = NEW.Email) THEN
    -- verifica se la combinazione di "NomePremio" e "EmailUtente"
esiste già nella tabella "Vincente". Se non esiste, viene eseguito
l'inserimento, altrimenti l'istruzione di inserimento viene ignorata.
    -- In questo modo si evita di inserire duplicati nella tabella
"Vincente".

        INSERT INTO Vincente(NomePremio, EmailUtente)
        VALUES (NomePremio_locale, NEW.Email);
    END IF;
END IF;
END WHILE;

CLOSE CurPremi;
END;
//
DELIMITER ;

/*Se viene inserito un premio e l'utente ha già i punti necessari, bisogna
inserire il premio tra quelli vinti dall'utente*/
DELIMITER //
CREATE TRIGGER AssegnaPremioAppenaInserito AFTER INSERT ON Premio
FOR EACH ROW
BEGIN
    DECLARE done BOOLEAN DEFAULT FALSE;
    DECLARE EmailUtente VARCHAR(255);
    DECLARE TotalebonusUtente float;
    DECLARE cur CURSOR FOR SELECT Email, Totalebonus FROM Utente;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO EmailUtente, TotalebonusUtente;
        IF (NEW.Puntinecessari <= TotalebonusUtente AND NOT done) THEN
            INSERT INTO Vincente(NomePremio, EmailUtente)
            VALUES (NEW.Nome, EmailUtente);
        END IF;
        IF done THEN
            LEAVE read_loop;
        END IF;
    END IF;

```

```

        END LOOP;

        CLOSE cur;
    END;
//
DELIMITER ;

/*Incrementa il totale bonus dell'utente di 0.5 quando accetta un
sondaggio*/
DELIMITER //
CREATE TRIGGER IncrementaTotalebonus AFTER UPDATE ON Invito
FOR EACH ROW
BEGIN
    IF (NEW.Esito = 'ACCETTATO' AND OLD.Esito <> 'ACCETTATO') THEN
        UPDATE Utente SET Totalebonus = Totalebonus + 0.5 WHERE Email =
NEW.EmailUtente; -- Email di utente comparata con Email di Invito (NEW)
    END IF;
END;
//
DELIMITER ;

/*
Utilizzare un trigger per incrementare di 1 unità il campo #numsondaggi
quando
un utente premium inserisce un nuovo sondaggio.
*/
DELIMITER //
CREATE TRIGGER IncrementaNumsondaggi AFTER INSERT ON Sondaggio
FOR EACH ROW -- La clausola FOR EACH ROW specifica che il trigger deve
essere eseguito per ogni riga che viene inserita nella tabella.
BEGIN
    UPDATE UtentePremium
    SET numsondaggi = numsondaggi + 1
    WHERE Email = NEW.EmailUtentecreante;
END;
//
DELIMITER ;

# STORED PROCEDURE
/*Registrazione utente*/

```



```
DELIMITER //
```

```
CREATE PROCEDURE InserisciUtente  
(IN Email varchar(255), Pwd varchar(30), Nome varchar(30), Cognome  
varchar(30), Annonascita timestamp, Luogonascita varchar(50), PAS ENUM  
( 'PREMIUM', 'AMMINISTRATORE', 'SEMPLICE' ))  
BEGIN  
    INSERT INTO Utente(Email, Pwd, Nome, Cognome, Annonascita,  
Luogonascita, Totalebonus, PAS) VALUES  
        (Email, Pwd, Nome, Cognome, Annonascita, Luogonascita, 0, PAS);  
END  
//  
DELIMITER ;
```

```
/*Utente semplice diventa premium*/  
DELIMITER //  
CREATE PROCEDURE DiventaPremium  
(IN Email_parametro varchar(255))  
BEGIN  
    UPDATE Utente  
    SET PAS = 'PREMIUM'  
    WHERE Email = Email_parametro;  
END  
//  
DELIMITER ;
```

```
/*Utente semplice diventa amministratore*/  
DELIMITER //  
CREATE PROCEDURE DiventaAmministratore  
(IN Email_parametro varchar(255))  
BEGIN  
    UPDATE Utente  
    SET PAS = 'AMMINISTRATORE'  
    WHERE Email = Email_parametro;  
END  
//  
DELIMITER ;
```

```
/*Mostra i sondaggi*/  
DELIMITER //  
CREATE PROCEDURE MostraSondaggi
```

```

>EmailUtentecreante_parametro varchar(255), CFAziendacreante_parametro
varchar(16))
BEGIN
    SELECT * FROM Sondaggio WHERE EmailUtentecreante =
EmailUtentecreante_parametro OR CFAziendacreante =
CFAziendacreante_parametro;
END
//
DELIMITER ;

/*Mostra domini*/
DELIMITER //
CREATE PROCEDURE MostraDomini
()
BEGIN
    SELECT * FROM Dominio;
END
//
DELIMITER ;

/*Inserisci interesse dell'utente per un dominio*/
DELIMITER //
CREATE PROCEDURE InserisciInteresse
(EmailUtente varchar(255), ParolachiaveDominio varchar(255))
BEGIN
    INSERT INTO Interessato(EmailUtente, ParolachiaveDominio) VALUES
(EmailUtente, ParolachiaveDominio);
END
//
DELIMITER ;

/*Rimuovi interesse dell'utente per un dominio*/
DELIMITER //
CREATE PROCEDURE RimuoviInteresse
(EmailUtente_parametro varchar(255), ParolachiaveDominio_parametro
varchar(255))
BEGIN
    DELETE FROM Interessato WHERE (EmailUtente = EmailUtente_parametro
AND ParolachiaveDominio = ParolachiaveDominio_parametro);
END

```

```

//
DELIMITER ;

/*Mostra gli inviti dell'utente comprendendo le informazioni del sondaggio
di riferimento*/
DELIMITER //
CREATE PROCEDURE MostraInvitiUtente
(EmailUtente_parametro varchar(255))
BEGIN
    SELECT Invito.ID, Invito.Esito, Sondaggio.Titolo,
Sondaggio.DataCreazione, Sondaggio.DataChiusura,
Sondaggio.ParolachiaveDominio
    FROM Invito JOIN Sondaggio ON Invito.CodiceSondaggio=Sondaggio.Codice
    WHERE Invito.EmailUtente = EmailUtente_parametro;
END
//
DELIMITER ;

/*Accetta/rifiuta invito: fa update dell'esito dell'invito settando a
ACCETTATO o RIFIUTATO in base all'input della stored procedure*/
DELIMITER //
CREATE PROCEDURE AccettaRifiutaInvito
(Decisione boolean, /*1 -> true, 0 -> false*/ EmailUtente_parametro
varchar(255), ID_parametro integer)
BEGIN
    IF (Decisione = 0) THEN
        UPDATE Invito
        SET Esito = "RIFIUTATO"
        WHERE EmailUtente = EmailUtente_parametro AND ID =
ID_parametro;
    ELSEIF (Decisione = 1) THEN
        UPDATE Invito
        SET Esito = "ACCETTATO"
        WHERE EmailUtente = EmailUtente_parametro AND ID = ID_parametro;
    END IF;
END//
DELIMITER ;

```

```

/*Mostra la lista di sondaggi accettati dall'utente dato in input*/
DELIMITER //
CREATE PROCEDURE MostraSondaggiAccettati
(EmailUtente_parametro varchar(255))
BEGIN
    SELECT *
    FROM Invito JOIN Sondaggio ON Invito.CodiceSondaggio = Sondaggio.Codice
    WHERE EmailUtente = EmailUtente_parametro AND Esito = 'ACCETTATO';
END
//
DELIMITER ;

/*Inserisce la risposta alla domanda aperta*/
DELIMITER //
CREATE PROCEDURE InserisciRisposta
(Testo text, IDDomandaaperta integer, EmailUtente varchar(255))
BEGIN
    INSERT INTO Risposta(Testo, IDDomandaaperta, EmailUtente) VALUES
    (Testo, IDDomandaaperta, EmailUtente);
END
//
DELIMITER ;

/*Inserisce l'opzione selezionata per la domanda chiusa*/
DELIMITER //
CREATE PROCEDURE InserisciOpzioneRisposta
(EmailUtente varchar(255), IDDomandachiusaOpzione integer,
NumeroprogressivoOpzione integer)
BEGIN
    INSERT INTO SelezionanteUtenteOpzione(EmailUtente,
IDDomandachiusaOpzione, NumeroprogressivoOpzione) VALUES
    (EmailUtente, IDDomandachiusaOpzione, NumeroprogressivoOpzione);
END
//
DELIMITER ;

/*Mostra risposta in base all'utente dato in input e all'id della domanda
dato in input*/
DELIMITER //
CREATE PROCEDURE MostraRispostaAperta

```

```

(EmailUtente_parametro varchar(255), IDDomandaaperta_parametro integer)
BEGIN
    SELECT * FROM Risposta WHERE EmailUtente = EmailUtente_parametro AND
IDDomandaaperta = IDDomandaaperta_parametro;
END
//
DELIMITER ;

/*Mostra le risposte alla domanda aperta data in input*/
DELIMITER //
CREATE PROCEDURE MostraRisposte
(IDDomandaaperta_parametro integer)
BEGIN
    SELECT * FROM Risposta WHERE IDDomandaaperta =
IDDomandaaperta_parametro;
END
//
DELIMITER ;

/*Mostra l'opzione selezionata dall'utente dato in input come risposta alla
domanda il cui id e' dato in input*/
DELIMITER //
CREATE PROCEDURE MostraOpzioneSelezionata
(EmailUtente_parametro varchar(255), IDDomandachiusa_parametro integer)
BEGIN
    SELECT *
    FROM SelezionanteUtenteOpzione JOIN Opzione ON IDDomandachiusaOpzione =
IDDomandachiusa AND NumeroprogressivoOpzione = Numeroprogressivo
    WHERE EmailUtente = EmailUtente_parametro AND IDDomandachiusa =
IDDomandachiusa_parametro;
END
//
DELIMITER ;

/*Mostra le opzioni non selezionate dall'utente dato in input per la domanda
chiusa data in input*/
DELIMITER //
CREATE PROCEDURE MostraOpzioniNonSelezionate(EmailUtente_parametro
varchar(255), IDDomandachiusa_parametro integer)
BEGIN

```

```

SELECT IDDomandachiusa, Numeroprogressivo, Testo
FROM Opzione
WHERE IDDomandachiusa = IDDomandachiusa_parametro

AND (IDDomandachiusa, Numeroprogressivo) NOT IN (
    SELECT IDDomandachiusaOpzione, NumeroprogressivoOpzione
    FROM SelezionanteUtenteOpzione
    WHERE EmailUtente = EmailUtente_parametro AND IDDomandachiusa =
IDDomandachiusa_parametro
);
END//
DELIMITER ;

```

```

/*Mostra se l'utente dato in input ha risposto alle domande aperte per il
sondaggio dato in input*/

```

```

DELIMITER //
CREATE PROCEDURE MostraRisposteDomandeAperteSondaggio
(EmailUtente_parametro varchar(255), CodiceSondaggio_parametro integer)
BEGIN
    SELECT *
    FROM Risposta JOIN ComponenteSondaggioDomanda ON IDDomandaaperta =
IDDomanda
    WHERE EmailUtente = EmailUtente_parametro AND CodiceSondaggio =
CodiceSondaggio_parametro;
END
//
DELIMITER ;

```

```

/*Mostra se ci sono risposte (opzioni selezionate per domande chiuse) per il
sondaggio dato in input*/

```

```

DELIMITER //
CREATE PROCEDURE MostraOpzioniDomandeChiuseSondaggio
(EmailUtente_parametro varchar(255), CodiceSondaggio_parametro integer)
BEGIN
    SELECT *
    FROM SelezionanteUtenteOpzione JOIN ComponenteSondaggioDomanda ON
IDDomandachiusaOpzione = IDDomanda
    WHERE EmailUtente = EmailUtente_parametro AND CodiceSondaggio =
CodiceSondaggio_parametro;
END

```

```

//
DELIMITER ;

/*Mostra utenti che hanno risposto al sondaggio dato in input*/
DELIMITER //
CREATE PROCEDURE MostraUtentiCheHannoRisposto
(IDDomanda_parametro integer, CodiceSondaggio_parametro integer)
-- posso usare lo stesso iddomanda in quanto ogni domanda ha un id, e quindi
una domanda aperta e chiusa con stesso id non possono esistere
BEGIN
    -- Qui prendo gli utenti che hanno risposto alle domande aperte del
sondaggio (se ce ne sono)
    SELECT EmailUtente
    FROM Risposta
    JOIN ComponenteSondaggioDomanda ON IDDomandaaperta = IDDomanda
    WHERE IDDomandaaperta = IDDomanda_parametro AND CodiceSondaggio =
CodiceSondaggio_parametro
    UNION
    -- Qui prendo gli utenti che hanno risposto alle domande chiuse del
sondaggio (se ce ne sono)
    SELECT EmailUtente
    FROM SelezionanteUtenteOpzione
    JOIN ComponenteSondaggioDomanda ON IDDomandachiusaOpzione =
IDDomanda
    WHERE IDDomandachiusaOpzione = IDDomanda_parametro AND
CodiceSondaggio = CodiceSondaggio_parametro;
    -- Se questa query ritorna nessuna riga vuol dire che nessuno ha
ancora risposto al sondaggio
END
//
DELIMITER ;

/*STATISTICHE*/

/*Visualizza utenti in ordine di punteggio max -> min*/
DELIMITER //
CREATE PROCEDURE VisualizzaClassifica
()
BEGIN
    SELECT Email, Totalebonus FROM Utente ORDER BY Totalebonus DESC;

```

```

END
//
DELIMITER ;

/*Visualizza tutti i premi*/
DELIMITER //
CREATE PROCEDURE VisualizzaPremi
()
BEGIN
    SELECT * FROM Premio ORDER BY Puntinecessari DESC;
END
//
DELIMITER ;

/*Inserisce un nuovo premio*/
DELIMITER //
CREATE PROCEDURE InserisciPremio
(Nome varchar(255), Descrizione text, Foto longblob, Puntinecessari float,
EmailUtenteAmministratore varchar(255))
BEGIN
    INSERT INTO Premio(Nome, Descrizione, Foto, Puntinecessari,
EmailUtenteAmministratore) VALUES
    (Nome, Descrizione, Foto, Puntinecessari,
EmailUtenteAmministratore);

END//
DELIMITER ;

/*Inserisce un nuovo dominio*/
DELIMITER //
CREATE PROCEDURE InserisciDominio
(Parolachiave varchar(255), Descrizione text)
BEGIN
    INSERT INTO Dominio(Parolachiave, Descrizione) VALUES
    (Parolachiave, Descrizione);
END
//
DELIMITER ;

```



```

/*Crea un nuovo sondaggio*/
DELIMITER //
CREATE PROCEDURE CreaSondaggio
(Titolo varchar(255), Stato ENUM ('APERTO', 'CHIUSO'), MaxUtenti integer,
DataCreazione timestamp, DataChiusura timestamp, ParolachiaveDominio
varchar(255), CFAziendacreante varchar(16), EmailUtentecreante varchar(255))
BEGIN
    INSERT INTO Sondaggio(Titolo, Stato, MaxUtenti, DataCreazione,
DataChiusura, ParolachiaveDominio, CFAziendacreante, EmailUtentecreante)
VALUES
    (Titolo, Stato, MaxUtenti, DataCreazione, DataChiusura,
ParolachiaveDominio, CFAziendacreante, EmailUtentecreante);
END
//
DELIMITER ;

/*Mostra gli utenti potenziali per essere invitati in automatico
dall'azienda per lo specifico sondaggio*/
DELIMITER //
CREATE PROCEDURE MostraUtentiInteressati
(CodiceSondaggio_parametro integer)
BEGIN
    SELECT * FROM Sondaggio JOIN Interessato on
Sondaggio.ParolachiaveDominio = Interessato.ParolachiaveDominio WHERE
Sondaggio.Codice = CodiceSondaggio_parametro;
END
//
DELIMITER ;

/*Mostra utenti interessati ad un determinato sondaggio in base al dominio
di interesse e che non abbiano già l'invito per quel sondaggio*/
DELIMITER //
CREATE PROCEDURE MostraUtentiInteressatiSenzaInvito
(ParolachiaveDominio_parametro varchar(255), CodiceSondaggio_parametro
integer)
BEGIN
    SELECT Utente.Email, Utente.Nome, Utente.Cognome, Utente.Annonascita,
Utente.Luogonascita
    FROM Utente JOIN Interessato ON Utente.Email=Interessato.EmailUtente
JOIN Dominio ON Interessato.ParolachiaveDominio=Dominio.ParolaChiave JOIN

```

```

Sondaggio ON Dominio.ParolaChiave=Sondaggio.ParolachiaveDominio
    WHERE Dominio.ParolaChiave = ParolachiaveDominio_parametro AND
Sondaggio.Codice = CodiceSondaggio_parametro
    AND Utente.Email NOT IN
        (SELECT EmailUtente
         FROM Invito
         WHERE CodiceSondaggio = CodiceSondaggio_parametro);
END
//
DELIMITER ;

/*Inserisci un invito per un utente*/
DELIMITER //
CREATE PROCEDURE InserisciInvito
(EmailUtente varchar(255), CodiceSondaggio integer, CFAziendainvitante
varchar(16), EmailUtenteinvitante varchar(255))
BEGIN
    INSERT INTO Invito(Esito, EmailUtente, CodiceSondaggio,
CFAziendainvitante, EmailUtenteinvitante) VALUES
        ('SOSPESO', EmailUtente, CodiceSondaggio, CFAziendainvitante,
EmailUtenteinvitante);
END
//
DELIMITER ;

/*Inserisce una nuova domanda, e se APERTA inserisce in DomandaAperta*/
DELIMITER //
CREATE PROCEDURE InserisciDomanda
(Testo varchar(3000), Foto longblob, Punteggio integer, ApertaChiusa ENUM
('APERTA', 'CHIUSA'), CFAziendainserente varchar(16), EmailUtenteinserente
varchar(255), MaxCaratteriRisposta_parametro integer, CodiceSondaggio
integer)
BEGIN
    DECLARE ultimoID integer;

    INSERT INTO Domanda(Testo, Foto, Punteggio, ApertaChiusa,
CFAziendainserente, EmailUtenteinserente) VALUES
        (Testo, Foto, Punteggio, ApertaChiusa, CFAziendainserente,
EmailUtenteinserente);

```

```

        SET ultimoID = LAST_INSERT_ID();

        INSERT INTO ComponenteSondaggioDomanda(CodiceSondaggio, IDDomanda)
VALUES
        (CodiceSondaggio, ultimoID);

        IF (ApertaChiusa = "APERTA") THEN
        UPDATE DomandaAperta
        SET MaxCaratteriRisposta = MaxCaratteriRisposta_parametro
        WHERE ID = ultimoID;
        END IF;
END
//
DELIMITER ;

/*Mostra le domande in base al codice del sondaggio dato in input*/
DELIMITER //
CREATE PROCEDURE MostraDomande
(CodiceSondaggio_parametro integer)
BEGIN
        SELECT * FROM Domanda JOIN ComponenteSondaggioDomanda ON Domanda.ID
= ComponenteSondaggioDomanda.IDDomanda WHERE CodiceSondaggio =
CodiceSondaggio_parametro;
END
//
DELIMITER ;

/*Mostra le opzioni per il sondaggio dato in input*/
DELIMITER //
CREATE PROCEDURE MostraOpzioni
(IDDomandachiusa_parametro integer)
BEGIN
        SELECT * FROM Opzione WHERE IDDomandachiusa =
IDDomandachiusa_parametro;
END
//
DELIMITER ;

/*Aggiungi un'opzione per la domanda data in input*/
DELIMITER //

```

```

CREATE PROCEDURE InserisciOpzione(IDDomandachiusa_parametro integer,
Testo_parametro text)
BEGIN

    INSERT INTO Opzione (IDDomandachiusa, Testo) VALUES
    (IDDomandachiusa_parametro, Testo_parametro);

END//
DELIMITER ;

/*Elimina l'opzione data in input per la domanda data in input*/
# Nel momento in cui si rimuove, devo andare a ricalcolare i numeri
progressivi delle opzioni in base all'id della domanda chiusa
DELIMITER //
CREATE PROCEDURE RimuoviOpzione(IDDomandachiusa_parametro INTEGER,
Numeroprogressivo_parametro TEXT)
BEGIN
    -- Eseguo la rimozione dell'opzione con il parametro passato
    DELETE FROM Opzione WHERE IDDomandachiusa = IDDomandachiusa_parametro
AND Numeroprogressivo = Numeroprogressivo_parametro;

    -- Aggiorno i progressivi delle opzioni rimanenti dopo la rimozione
dell'opzione
    /*Decremento "Numeroprogressivo" di 1 per ogni riga che ha un
"IDDomandachiusa"
    uguale a quello della riga eliminata e un "Numeroprogressivo"
maggiore di quello della riga eliminata.
    In questo modo gli indici progressivi saranno automaticamente
aggiornati ad ogni eliminazione,
    evitando "salti" tra le opzioni rimanenti*/
    UPDATE Opzione
    SET Numeroprogressivo = Numeroprogressivo - 1
    WHERE IDDomandachiusa = IDDomandachiusa_parametro AND Numeroprogressivo
> Numeroprogressivo_parametro;
END//
DELIMITER ;

/*Conta il numero di risposte alla domanda aperta data in input*/
DELIMITER //
CREATE PROCEDURE ContaNumeroRisposteDomandaAperta

```

```

(IDDomandaaperta_parametro integer)
BEGIN
    SELECT COUNT(*) AS NumeroRisposte
    FROM Risposta
    WHERE IDDomandaaperta = IDDomandaaperta_parametro;
END
//
DELIMITER ;

/*Conta il numero di risposte alla domanda aperta data in input*/
DELIMITER //
CREATE PROCEDURE ContaNumeroRisposteDomandaChiusa
(IDDomandachiusaOpzione_parametro integer)
BEGIN
    SELECT COUNT(*) AS NumeroRisposte
    FROM SelezionanteUtenteOpzione
    WHERE IDDomandachiusaOpzione = IDDomandachiusaOpzione_parametro;
END
//
DELIMITER ;

/*Conta il numero di occorrenze per una specifica opzione di uno specifico
sondaggio*/
DELIMITER //
CREATE PROCEDURE ContaNumeroOccorrenzeOpzione
(IDDomandachiusaOpzione_parametro integer,
NumeroprogressivoOpzione_parametro integer)
BEGIN
    SELECT COUNT(*) AS NumeroOccorrenze
    FROM SelezionanteUtenteOpzione
    WHERE IDDomandachiusaOpzione = IDDomandachiusaOpzione_parametro AND
NumeroprogressivoOpzione = NumeroprogressivoOpzione_parametro;
END
//
DELIMITER ;

/*Elimina la domanda data in input*/
DELIMITER //
CREATE PROCEDURE RimuoviDomanda(ID_parametro INTEGER)
BEGIN

```

```
        DELETE FROM Domanda WHERE (ID = ID_parametro);
END//
DELIMITER ;

/*Elimina il sondaggio dato in input*/
DELIMITER //
CREATE PROCEDURE EliminaSondaggio(Codice_parametro INTEGER)
BEGIN
        DELETE FROM Sondaggio WHERE (Codice = Codice_parametro);
END//
DELIMITER ;

/*Registrazione azienda*/
DELIMITER //
CREATE PROCEDURE InserisciAzienda(CF varchar(16), Pwd varchar(30), Email
varchar(255), Nome text, Sede text)
BEGIN
        INSERT INTO Azienda (CF, Pwd, Email, Nome, Sede) VALUES
        (CF, Pwd, Email, Nome, Sede);
END//
DELIMITER ;
```